NAME

EE431 Quiz No. 2: micropro

Date: Tuesday, November 20, 2001
Time = 2.5 hours
Written material of any sort and computer disks allowed

1. Generate a schematic diagram for the Verilog descriptions given below. For the flip/flops in your schematic, use a D type with clock enable, asynchronous set and asynchronous clear.

(1) (a)
```
module circuit_1 (clk, a, b, c, d)
input clk, a;
output b, c, d;
reg b, c, d;
always @ (posedge clk)
begin
b = a;
c = b;
end
always @ (posedge clk)
d = c;
```

15

(1) (b)
```
module circuit_2 (clk, a, b, c, d)
input clk, a;
output b, c, d;
reg b, c, d;
always @ (posedge clk)
begin
b <= a;
c <= b;
end
always @ (posedge clk)
d <= c;
```

(1) (c)
```
module circuit_3 (clk, a, b, c, d)
input clk, a;
output b, c, d;
reg b, c, d;
always @ (posedge clk)
```

```
begin
b = a;
c = b;
end
always @ (a or c or b)
if (b==c) d = a;
else d = ~a;
```

(1) (d)

```
module circuit_4 (clk, a, b, c, d)
input clk, a;
output b, c, d;
reg b, c, d;
always @ (posedge clk)
begin
if (c==d) begin b = a; c = b; end
else begin b = c; c = d; end
end
always @ (posedge clk)
d = c;
```

2. Consider the structural description below

```
module circuit_5(clk, a, b, c, d)
input clk, a;
output b, c, d;
DFFE flop_1 (.clk(clk), .d(~b), .PRN(a), .Q(b));
DFFE flop_2 (.clk(clk), .d(~c), .ENA(b), .Q(c));
assign d = a| c;
endmodule

DESCRIPTION OF CONNECTION LIST FOR D FLOP
DFFE instance_name (
   .D(d),         // the d input
   .CLK(clk),     // the clock, positive edge trigger
   .ENA(enable), // if used, D goes to Q on clock edge when
                  // ENA is high  otherwise Q does not change
   .CLRN(clear_bar), // if used, clears Q when low
   .PRN(set_bar),    // if used, sets Q when low
```

```
    .Q(q)     // output
    );
```

(2) (a) Write a behavioral description in Verilog of the circuit described above. For this description use only blocking equalities in the "always" constructs.

(2) (b) Write a behavioral description in Verilog of the circuit described above. For this description use only non-blocking equalities in the "always" constructs.

(15) 3. This is the Lab portion of the exam. You will be required to modify your microprocessor, instantiate it inside a verilog HDL (provided), simulate the resulting circuit and then report results at key times.

Revise you microprocessor as follows:

(a) Make the program counter, which is described in the program sequencer, an output of the microprocessor. Call that output pc.

(b) Facilitate a two bit input to the micro processor called 'jam_address', i.e.'jam_address[1:0]'. This two bit input is to be connected inside the program sequencer to the next address logic block. The program sequencer (next address logic block) is to modified to behave as follows:

i. If jam_address[1:0] == 2'b00, the program sequencer runs as it does now.

ii. If jam_address[1:0] == 2'b01, the program sequencer ignores all other inputs and makes pm_address = 8'b0000_0000.

iii. If jam_address[1:0] == 2'b10, the program sequencer ignores all other inputs and makes pm_address = 8'b1000_0000.

iv. If jam_address[1:0] == 2'b11, the program sequencer ignores all other inputs and makes pm_address = 8'b1100_0000.

(c) Instantiate your micro inside the verilog HDL module called 'quiz_1_2001' (found in file G:
classes
ee431

quiz_1_2001.v) by modifying the example instantiation that is inside the module. A hex file containing the program your micro is to run is found in the same directory and has the name 'quiz_1_2001_rom.hex'

(d) Compile and test_micro for a FLEX10K part, preferably a 20,000 gate part, i.e. a FELX10k20$\cdots$. Do whatever you have to do to get the program in the file 'quiz_1_2001_rom.hex' into your program memory rom.

(e) Module 'quiz_1_2001' has two 1 bit inputs called 'clk' and 'test_control', a 4 bit input called 'input_code' and two outputs. The outputs are 'timer', which is 8 bits and 'output_code', which is 12 bits.

With the help of the waveform editor simulate the circuit. Set the end time of your simulation to 128 micro seconds (clock periods). Make input 'clk' a square clock with a period of 1 microsecond. The input 'input_code' must be set to the value shown in the table on the next page. If you make input 'test_control' 1'b1 for the entire 128 microsecond (and your micro is working properly) the 'output_code' will have the values specified in the table.

****
To test your modifications
****
if input 'input_code' is set to the constant 4'H0
and if test_control is the constant 1'b1

timer	output_code
8'H30	12'H206
8'H5B	12'HA0A
8'H75	12'HD07

****
To get results for your exam
****
Set 'test_control' to the constant 1'b0, and

set input_code' to the constant 4'H5

and report the value of 'output_code' in hex

for the times listed below:

timer	output_code
8'H30	12'H751
8'H5B	12'HE50
8'H75	12'H85F